# Python GUI程序设计

张晓 西北工业大学计算机学院
zhangxiao@nwpu.edu.cn
2009-11-20

# Python GUI编程概述

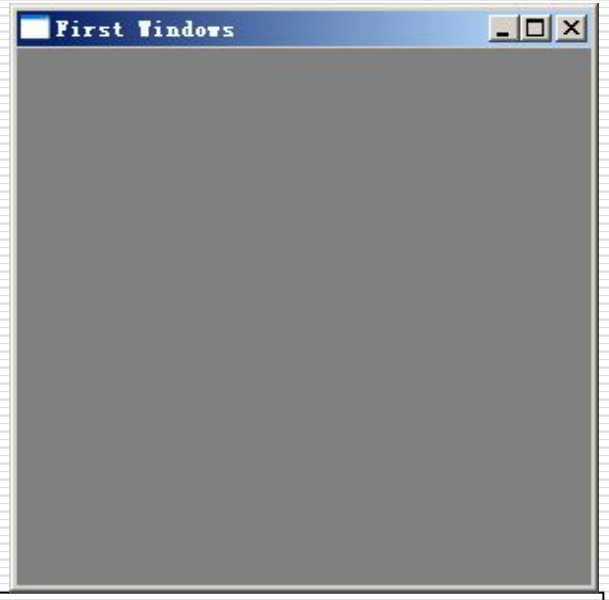
- PyQT
- wxPython
- wxPython参考资料
  - 活学活用wxPython
  - http://www.wxpython.org/

# wxPython是什么

- wxPython 实际是两件事物的组合：Python 脚本语言和 GUI 功能的 wxWindows 库
- wxWindows库是为了最大可移植性的 C/C++ 库，而抽取 GUI 功能
  - Windows
  - X、KDE 或 Gnome 的 UNIX
  - wxPython 应用程序不仅快速和易于编写，而且可以在不作任何更改情况下，运行在 Windows 或 UNIX 环境下
- http://www.wxpython.org/

# 最小的 wxPython 程序

- 显示一个空白的窗口，标题为First Windows
- App对象
- Frame对象




```
import wx

app = wx.PySimpleApp()
frame = wx.Frame(None, -1, "First Windows", size=(300, 300))
frame.Show(True)

app.MainLoop()
```

# 显示JPG图片

- 使用image对象
- 将image对象传递给frame


```
class Frame(wx.Frame):   #2 wx.Frame子类
    """Frame class that displays an image."""

    def __init__(self, image, parent=None, id=-1,
                 pos=wx.DefaultPosition,
                 title='Hello, wxPython!'): #3图像参数
        """Create a Frame instance and display image."""
        #4 显示图像
        temp = image.ConvertToBitmap()
        size = temp.GetWidth(), temp.GetHeight()
        wx.Frame.__init__(self, parent, id, title, pos, size)
        self.bmp = wx.StaticBitmap(parent=self, bitmap=temp)
```

# 图形设计的基础

- 应用程序
- 窗口
- 控件
- 事件处理

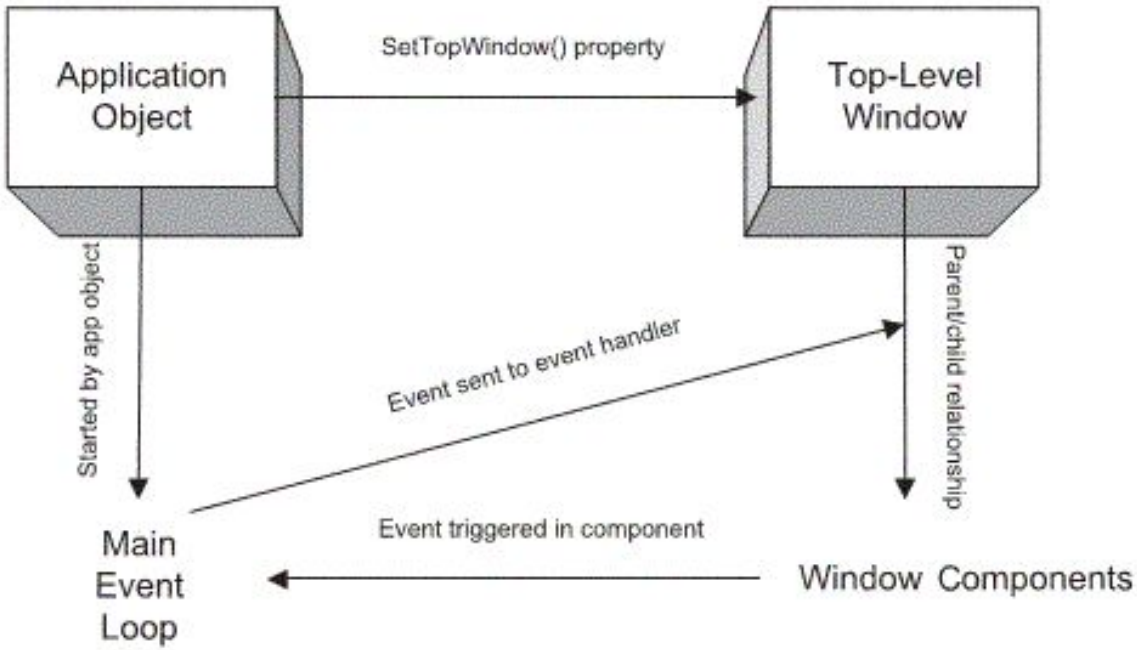



**Figure 2.1 A schematic of the basic wxPython application structure, showing the relationship between the application object, the top-level window, and the main event loop**

# 各对象的生命周期

- 应用程序
- 窗口/控件
- 事件处理

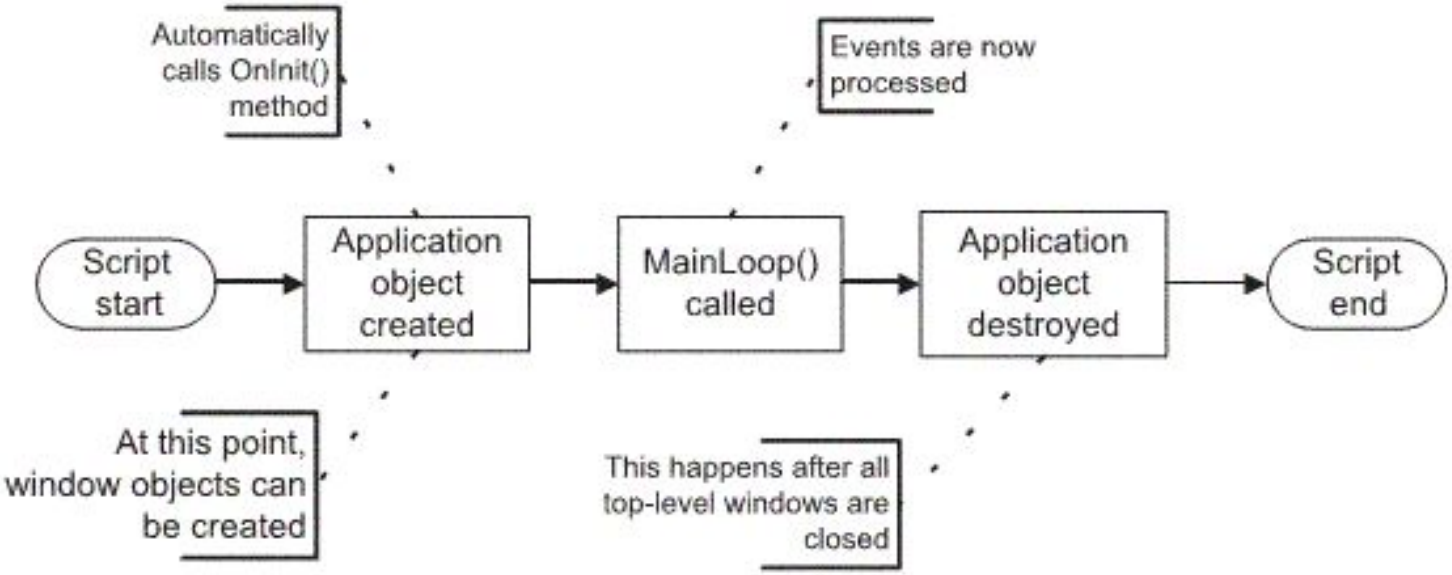



**Figure 2.2 Major events in the wxPython application lifecycle, including the beginning and ending of both the wxPython application and the script which surrounds it**

# 增加一个控件

- 创建并修改状态栏

```
statusBar = self.CreateStatusBar() #1 创建状态栏
self.SetStatusText("This is the statusbar")
```

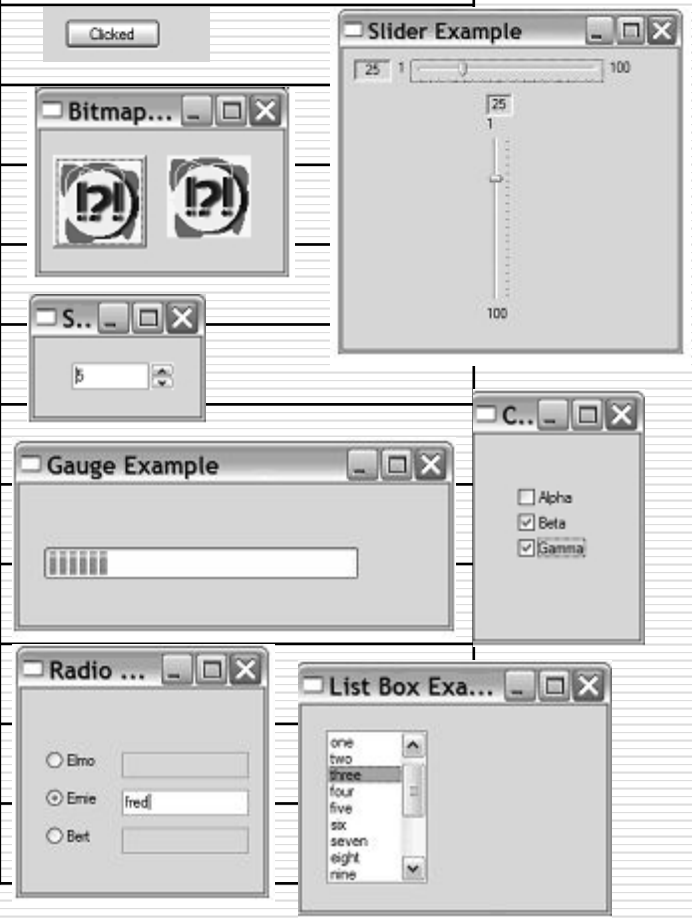
- 增加一个Menu

```
 menuBar = wx.MenuBar() # 创建菜单栏
# 创建两个菜单
 menu1 = wx.Menu("")
 menuBar.Append(menu1, "File")
self.SetMenuBar(menuBar)  # 在框架上附上菜单栏
```

- 增加静态文本，增加单选框，复选框，进度条

# 有哪些控件—wxPython Demo

| | |
|---|---|
| 静态文本 | wx.StaticText |
| 可编辑的文本 | wx.TextCtrl |
| 按钮 | wx.Button |
| 位图按钮 | wx.BitmapButton |
| 滑块 | wx.Slider |
| 微调 | wx.SpinCtrl |
| 进度条 | wx.Gauge |
| 复选框 | wx.CheckBox |
| 单选按钮 | wx.RadioButton |
| 选择器 | wx.Choice |
| 列表框 | wx.ListBox |
| 组合框和标尺 | wx.ComboBox |
| 合并复选框和列表框 | wx.CheckListBox |



# 事件驱动环境

- 事件处理是wxPython程序工作的基本机制。
- 事件就是发生在系统中的事，应用程序通过触发相应的功能以响应它。
  - 低级的用户动作，如鼠标移动或按键按下
  - 高级的用户动作（定义在wxPython的窗口部件中的），如单击按钮或菜单选择。
  - 系统动作，如关机。

# 事件驱动的术语

- 事件(event)：应用程序期间发生的事情，它要求有一个响应。
- 事件对象(event object)：在wxPython中，它具体代表一个事件，其中包括了事件的数据等属性。它是类wx.Event或其子类的实例，子类如wx.CommandEvent和wx.MouseEvent。
- 事件类型(event type)：wxPython分配给每个事件对象的一个整数ID。事件类型给出了关于该事件本身更多的信息。例如，wx.MouseEvent的事件类型标识了该事件是一个鼠标单击还是一个鼠标移动。
- 事件源(event source)：任何wxPython对象都能产生事件。例如按钮、菜单、列表框和任何别的窗口部件。
- 事件驱动(event-driven)：一个程序结构，它的大部分时间花在等待或响应事件上。
- 事件队列(event queue)：已发生的但未处理的事件的一个列表。
- 事件处理器(event handler)：响应事件时所调用的函数或方法。也称作处理器函数或处理器方法。
- 事件绑定器(event binder)：一个封装了特定窗口部件，特定事件类型和一个事件处理器的wxPython对象。为了被调用，所有事件处理器必须用一个事件绑定器注册。
- wx.EvtHandler：一个wxPython类，它允许它的实例在一个特定类型，一个事件源，和一个事件处理器之间创建绑定。注意，这个类与先前定义的事件处理函数或方法不是同一个东西。

# 事件驱动编程

- 在初始化设置之后，程序的大部分时间花在了一个空闲的循环之中
- 程序包含了对应于发生在程序环境中的事情的事件。
- 作为这个空闲的循环部分，程序定期检查是否有任何请求响应事情发生
- 当事件发生时，基于事件的系统试着确定相关代码来处理该事件，如果有，相关代码被执行

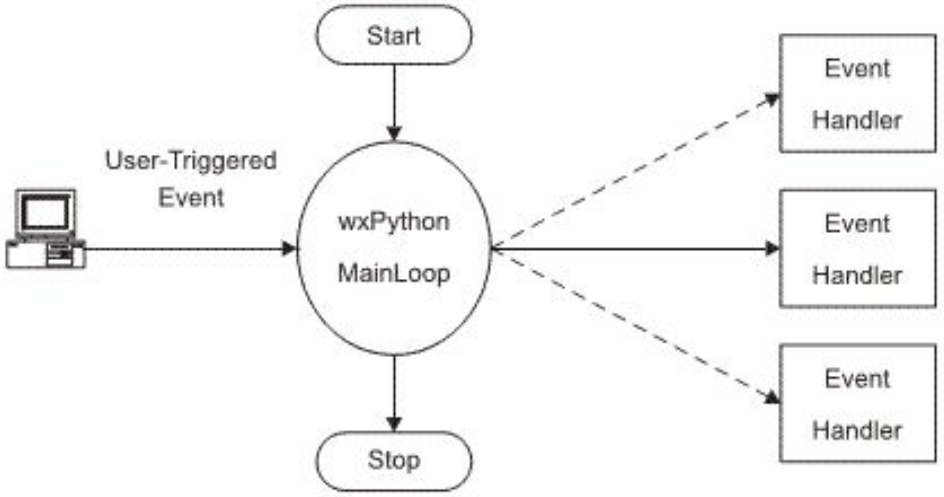



**Figure 3.1 A schematic of the event handling cycle, showing the life of the main program, a user event, and dispatch to handler functions.**

# 增加一个事件驱动的例子

- 定义一个事件处理函数

```
def OnMove(self, event):
    pos = event.GetPosition()
    self.SetStatusText("%s, %s" % (pos.x, pos.y))
```

- 将事件处理函数与窗体或控件关联，Bind

```
self.bmp.Bind(wx.EVT_MOTION,  self.OnMove)
```

```
Bind(event, handler, source=None, id=wx.ID_ANY, id2=wx.ID_ANY)
```

# 如何使用wxPython Demo中的例子

- Demo例子的结构
  - import wx #及其他需要的包
  - class TestPanel(wx.Panel):
  - def runTest(frame, nb, log):
  - Main函数(直接使用会失败)
- 定义一个log类处理输出
- 定义一个新的main
  - 创建一个app
  - 创建一个frame
  - 创建一个log
  - 调用RunTest函数

# 如何隐藏控制台窗口

- 可以用pythonw.exe启动程序，和python.exe的区别就是没有控制台窗口。
- 把文件扩展名改成.pyw也可以